由实例学习 TikZ 绘图 (1)
—— 机械手臂的绘制与标注

lyanry@gmail.com

August 31, 2007

这是我学习 TikZ 的一篇笔记，逐步讲解如何使用 TikZ 绘制图 1 所示机械手臂图形，其原始资料来源于 http://www.fauskes.net/pgftikzexamples/three-link-annotated/

1 作一些准备

首先要建立一个完整的 LaTeX 源文档环境，并在导言区中加载 tikz 宏包以及 patterns 库 (本文主要用该库绘制机器手臂底座处的阴影线)：

```
\documentclass{article}

\usepackage{tikz}
    \usetikzlibrary{patterns}
\usepackage{CJKutf8}

\begin{document}
\begin{CJK}{UTF8}{song}

\end{CJK}
\end{document}
```

使用 TikZ 绘图时，一定要尽力保证代码干净、易于维护与复用。图形中有可能要使用一些长度数值，为了修改与使用的方便，可以将它们定义为易于理解的变量形式。

定义带有单位的变量，可以使用 LaTeX 提供的 `\newlength` 与 `\setlength` 命令，它们可以定义一个新的长度命令并为其赋值。为了使用方便，将这两个命令封装为一个新变量 `\nvar`：

```
\newcommand{\nvar}[2]{
    \newlength{#1}
    \setlength{#1}{#2}
}
```

`\nvar` 命令接受两个必须的参数，参数 #1 为欲定义的长度命令，参数 #2 为长度命令的值。下面使用 `\nvar` 定义几个绘图变量：

```
\nvar{\dg}{0.3cm}
\nvar{\ddx}{1.5cm}
```

所定义的 \dg 为指定机械抓手宽度命令，\ddx 角度标注时所使用的引线长度命令。

使用 TEX 宏定义功能可以定义一些常数：

```
\def\dw{0.25}
\def\dh{0.5}
```

\dw 表示一个宽度值，\dh 分别表示一个高度值。

通过定义这种“变量”的表示形式，可以让你的绘图代码中不致于出现一堆不知所云的数字。

2 绘制底座

本节将定义一个专用于绘制机械手臂底座的 TEX 命令，这样做的目的主要是为了将来重用这段代码，如下：

```
\def\robotbase{
    \draw[rounded corners=8pt] (-\dw,-\dh)-- (-\dw, 0) --
        (0,\dh)--(\dw,0)--(\dw,-\dh);
    \draw (-0.5,-\dh)-- (0.5,-\dh);
    \fill[pattern=north east lines] (-0.5,-1) rectangle (0.5,-\dh);
}
```

\robotbase 宏的展开代码都是普通的 TikZ 路径绘制语句，只需要注意一下上一节定义的常数宏命令的使用。以后应当善于使用这种技巧，把你的代码从不知所云的数字中拯救出来。

使用 \robotbase 命令是很简单的，只需要在 tikzpicture 环境中调用即可：

```
\begin{tikzpicture}
    \robotbase
\end{tikzpicture}
```

绘制的机械手臂底座如图 1 所示：

图 1 机械手臂底座

3 绘制第一段手臂

本节主要定制两个 TEX 宏命令：\joint 与 \link，分别用于绘制关节与手臂。

```
\newcommand{\link}[2]{\draw [double distance=1.5mm, very thick]
    (0,0)--(#1:#2);}
\def\joint{
    \filldraw [fill=white] (0,0) circle (5pt);
    \fill[black] circle (2pt);
}
```

`\link` 命令可以绘制等宽双线，宽度为 1.5mm，起点为笛卡尔坐标系原点，其终点被设置为参数形式，可在使用该命令时指定。`\joint` 命令绘制了两个圆：大圆作为关节轮廓，采用白色填充；小圆作为关节与机械手臂的衔接点，采用黑色填充。

在已经绘制了机械手臂底座的 tikzpicture 环境中调用 `\link` 与 `\joint` 命令，绘制长度为 2cm，倾角为 30 度的第一段机械手臂，然后再绘制关节：

```
\begin{tikzpicture}
    \robotbase
    \link{30}{2cm}
    \joint
\end{tikzpicture}
```

对于代码中出现的 30 与 2cm 这样的数值，应当还是按照第一节中提到的原则，将其定义为宏命令来表示，因为这样的数值有可能在代码中要使用多次。再赘述一句，在使用 TikZ 绘图时，一定要保持清晰的认识，能快速反应出哪些数值需要用变量来表示。总的原则就是：你认为哪些数值对于你的图形而言是至关重要的，要多次使用的，那么就毫无疑问地将它们定义为 TEX 宏命令表示。因此对于第一段机械手臂的倾角与长度的宏命令定义如下：

```
\def\thetaone{30}
\def\Lone{2cm}
```

这样，tikzpicture 环境中的绘制机械手臂的代码就要作出相应替换了：

```
\begin{tikzpicture}
    \robotbase
    \link{\thetaone}{\Lone}
    \joint
\end{tikzpicture}
```

绘制结果如图 2 所示。

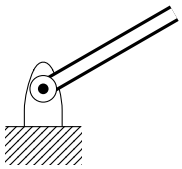

图 2　第一段手臂

4　绘制第二、三段手臂

由于已经有了绘制机械手臂与关节的命令了，所以绘制第二段手臂的任务变得尤为简单。基于上一节的 tikzpicture 环境，首先利用 TikZ 提供的坐标变换功能，将原来的坐标系变换为可以直接调用已定义的机械手臂与关节绘制命令的新坐标系，然后修改一下 `\link` 命令的倾角与长度参数即可绘制出第二段机械手臂。

由于在绘制第一段机械手臂时，其起点位置就是坐标原点，因此只需要将坐标原点移动到第一段机械手臂的末端，这样就可以将第二段机械手臂的起点定义在新的坐标原点位置了，这是因为在定义 `\link` 时，是将其起点定位在坐标原点的。

仅仅将坐标系平移到第一段机械手臂末端是不够的，还需要逆时针旋转 `\thetaone` 角度，让 x 轴与第一段机械手臂重合。这是因为 `link` 命令只能绘制与 x 重合的手臂图形。

TikZ 坐标系平移变换是通过设定路径参数 shift 的值来实现，坐标系旋转可设置 rotate 参数来实现，而且坐标变换只对当前路径有效。如果不想去修改 `\link` 与 `\joint` 的定义，但又想让它们都能在新的坐标系下工作，这就需要借助域 (scope) 环境来解决。

具体的坐标变换与第二段机械手臂绘制代码如下：

```
\begin{tikzpicture}
    \robotbase
    \link{\thetaone}{\Lone}
    \joint
    \begin{scope}[shift=(\thetaone:\Lone),rotate=\thetaone]
        \link{30}{2cm}
        \joint
    \end{scope}
\end{tikzpicture}
```

上面代码中，第二段手臂的倾角依然是 30 度，长度依然为 2cm，但是在坐标系变换的作用下，它的方位与第一段手臂是不同的。这里还要赘述一句，在坐标变换时，我们直接使用了 `\thetaone` 与 `\Lone` 这样的命令，这就是为什么前面一再强调重要数值要用宏命令来表示的主要原因。同样，对于第二段手臂的参数，依然应当使用这种宏命令表示方法：

```
\def\thetatwo{30}
\def\Ltwo{2cm}
```

使用同样的方法可以很轻易地绘制第三段手臂。

首先要定义第三段手臂的参数：

```
\def\thetathree{30}
\def\Lthree{2cm}
```

然后在绘制第二段手臂的域环境内进行坐标变换，并调用 `\link` 与 `\joint` 命令绘制第三段手臂。

本节所绘制的图形如图 3 所示。

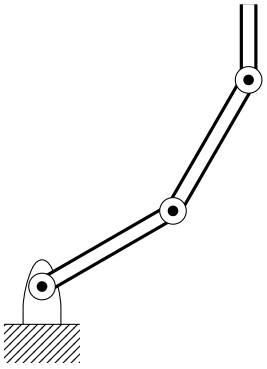

图 3　第二、三段手臂

5 绘制机械手

现在来定制一个绘制机械手的宏命令。在第 1 节中曾定义了一个长度命令 \dg，该命令所对应的长度值，在绘制机械手的命令定义中多次用到：

```
\def\grip{
    \draw[ultra thick](0cm,\dg)--(0cm,-\dg);
    \fill (0cm, 0.5\dg)+(0cm,1.5pt) -- +(0.6\dg,0cm) -- +(0pt,-1.5pt
        );
    \fill (0cm, -0.5\dg)+(0cm,1.5pt) -- +(0.6\dg,0cm) -- +(0pt,-1.5
        pt);
}
```

要注意的是 \grip 命令定义中，两个 \fill 命令用于绘制抓手，其中用到了坐标运算。\draw 命令用于绘制手腕部分。

在 tikzpicture 环境绘制第三段机械手臂的域环境中，再开辟一个新域，进行坐标变换，从而实现机械抓手的绘制：

```
\begin{tikzpicture}
    \robotbase
    \link(\thetaone:\Lone);
    \joint
    \begin{scope}[shift=(\thetaone:\Lone), rotate=\thetaone]
        \link(\thetatwo:\Ltwo);
        \joint
        \begin{scope}[shift=(\thetatwo:\Ltwo), rotate=\thetatwo]
            \link(\thetathree:\Lthree);
            \joint
            \begin{scope}[shift=(\thetathree:\Lthree),
                            rotate=\thetathree]
                \grip
            \end{scope}
        \end{scope}
    \end{scope}
\end{tikzpicture}
```

现在我们已经得到了机械手臂完整的图形，如图 4 所示。下一步要实现对该机械手臂的标注。

6 标注

对机械手臂的标注分为倾角标注与手臂长度标注。

先来解决角度标注的问题。下面定义一个专门用来绘制角度标注的命令，通过前面已经用过多次的坐标变换方法就可以实现这一命令的复用。

```
\newcommand{\angann}[2]{
    \begin{scope}[red]
        \draw [dashed, red] (0,0) -- (1.2\ddx,0pt);
```

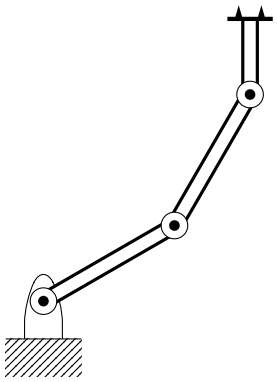

图 4　完整的机械手臂图

```
        \draw [->, shorten >=3.5pt] (\ddx,0pt) arc (0:#1:\ddx);
        \node at (#1/2-2:\ddx+8pt) {#2};
    \end{scope}
}
```

\angann 命令定义中，第一条 **\draw** 命令绘制一条角度引线，第二条 **\draw** 命令绘制表示角度大小且带有箭头的弧线。角度值的标注是使用一个结点来实现的。其实，理想的标注文本添加方式是在绘制弧线路径时直接添加，但很不幸，圆弧路径不支持结点自动放置，只好采用一种有点 dirty 的估算方法对结点进行定位。还应当注意的是在绘制弧线时，使用了一个 **shorten >** 参数，它的作用是指定弧线箭头与所接触的位置之间留出 3.5pt 的间隙，这样做的目的仅仅是为了美观。

在 tikzpicture 环境中调用 **\angann** 命令对机械手臂进行倾角标注的代码如下：

```
\begin{tikzpicture}
    \robotbase
     \angann{\thetaone}{$\theta_1$}
    \link{\thetaone}{\Lone}
    \joint
    \begin{scope}[shift=(\thetaone:\Lone), rotate=\thetaone]
        \angann{\thetatwo}{$\theta_2$}
        \link{\thetatwo}{\Ltwo}
        \joint
        \begin{scope}[shift=(\thetatwo:\Ltwo), rotate=\thetatwo]
            \angann{\thetathree}{$\theta_3$}
            \draw [dashed, red,rotate=\thetathree] (0,0) -- (1.2\ddx
                ,0pt);
            \link{\thetathree}{\Lthree}
            \joint
            \begin{scope}[shift=(\thetathree:\Lthree), rotate=\
                thetathree]
                 \grip
            \end{scope}
        \end{scope}
    \end{scope}
```

```
\end{tikzpicture}
```

上面代码已经不再神秘，需要注意的是进行第三段手臂倾角标注时，为了让标注美观，又绘制了一条辅助导引线。另外就是要注意标注代码的出现位置，要清楚应当是哪一种图元覆盖在哪一种图元之上，因为 TikZ 对于路径是按照绘制顺序进行覆盖的，而结点是待所有路径绘制完毕后才添加的。如果不明白我说啥，可以尝试调整一下倾角标注代码的出现位置，看看效果就明白了。

机械手臂倾角标注效果见图 5。

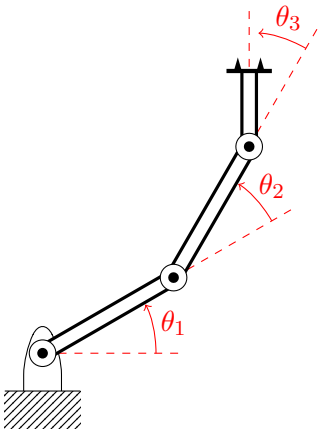

图 5　机械手臂倾角标注

与倾角标注一样，对于机械手臂的长度标注也是先定义一个专用命令，如下：

```
\newcommand{\lineann}[4][0.5]{
    \begin{scope}[rotate=#2, blue,inner sep=2pt]
        \draw[dashed, blue!40] (0,0) -- +(0,#1)
            node [coordinate, near end] (a) {};
        \draw[dashed, blue!40] (#3,0) -- +(0,#1)
            node [coordinate, near end] (b) {};
        \draw[|<->|] (a) -- node[fill=white] {#4} (b);
    \end{scope}
}
```

\lineann 命令是本文中参数最多的一个命令，它接受 4 个参数：标注线的偏移距离(可选)、标注线的倾角、标注线长度以及标注文本。

\lineann 命令定义中，开辟了一个域环境，并实施了一次坐标系旋转变换，旋转角度为参数 #2，目的是让标注线与机械手臂轴线平行。在域环境中绘制了三条路径，前两条路径是标注的引线，第三条路径是标注线。要认真体会一下前两条路径中的空结点用法，所谓空结点就是既无外围形状也无填充文本的结点，通过对其命名来表示一个特定位置。在第三条路径中，a 与 b 分别表示两条标注引线的末端位置。

还应当注意的是结点 a 与 b 都是 shape=coordinate 的结点，这种类型的结点只表示一个点位置，其路径的完整命令如下：

```
\path ... coordinate[ options ]( name )at( coordinate ) ... ;
```

在 tikzpicture 环境中调用长度标注命令时，依然要注意一下代码的出现位置，如下：

```
\begin{tikzpicture}
    \robotbase
    \angann{\thetaone}{$\theta_1$}
    \lineann[0.7]{\thetaone}{\Lone}{$L_1$}
    \link{\thetaone}{\Lone}
    \joint
    \begin{scope}[shift=(\thetaone:\Lone), rotate=\thetaone]
        \angann{\thetatwo}{$\theta_2$}
        \lineann[-1.5]{\thetatwo}{\Ltwo}{$L_2$}
        \link{\thetatwo}{\Ltwo}
        \joint
        \begin{scope}[shift=(\thetatwo:\Ltwo), rotate=\thetatwo]
            \angann{\thetathree}{$\theta_3$}
            \lineann[0.7]{\thetathree}{\Lthree}{$L_3$}
            \draw [dashed, red,rotate=\thetathree] (0,0) -- (1.2\ddx
                ,0pt);
            \link{\thetathree}{\Lthree}
            \joint
            \begin{scope}[shift=(\thetathree:\Lthree), rotate=\
                thetathree]
                \grip
            \end{scope}
        \end{scope}
    \end{scope}
\end{tikzpicture}
```

现在，一切都结束了，来欣赏一下吧，看图 6。

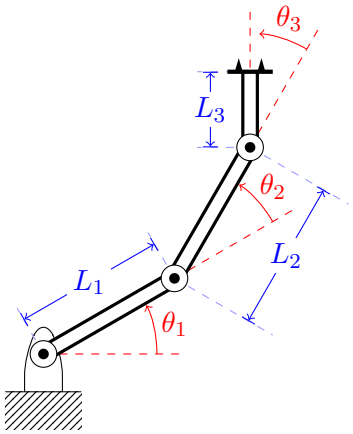

图 6　一切都 Over 了

结语

用 TikZ 绘图有些累人，但对于带有量度的精确图而言，即便采用所见即所得的绘图工具，也并不见得在效率上能有所提升。

使用 TikZ，熟悉常用语法固然重要，但更重要的是有程序设计的思维，尽量将绘图任务分解为模块，每个模块定义一个命令来实现，最后在 tikzpicture 环境中将这些模块组装起来。对于图形中诸多重复使用的数值，可以定义为长度命令或 TEX 宏来表示。要经常问自己：我写的 TikZ 代码干净么？